1.Abstract

The paper illustrates the implementation of a concurrent data-flow image processing application on multiprocessor and single processor (with and without RTOS) respectively. The main focus is how to take advantage of shared resources and schedule task in real time operating system. By measuring the execution time and memory footprint, we can compare the difference between bare-metal implementation and RTOS implementation.

2.Image processing

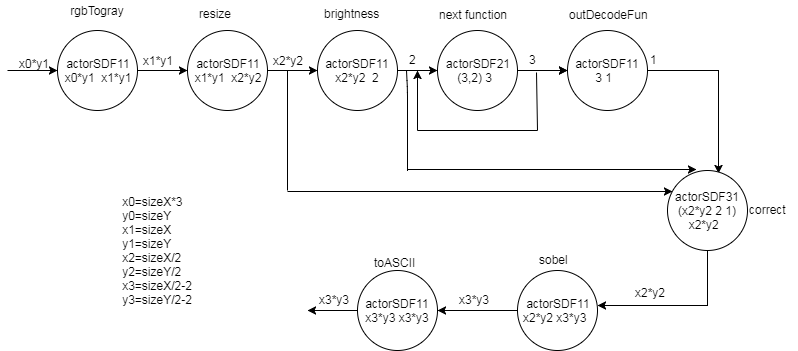


Fig 1 Synchronous Data Flow of Image Processing

The image processing application is developed to process a cycle of 10 different images as continuously input. As displayed in the synchronous data flow graph, the following steps must be done in order to get a processed pictures.

1. Converts an image of pixels into its grayscale

Grayscale [image](https://en.wikipedia.org/wiki/Image" \o "Image) is one in which the value of each [pixel](https://en.wikipedia.org/wiki/Pixel" \o "Pixel) is a single [sample](https://en.wikipedia.org/wiki/Sample_(signal)" \o "Sample (signal)) representing only an amount of light, that is, it carries only [intensity](https://en.wikipedia.org/wiki/Luminous_intensity" \o "Luminous intensity) information. Images of this sort are composed exclusively of shades of [gray](https://en.wikipedia.org/wiki/Grey" \o "Grey), varying from black at the weakest intensity to white at the strongest. And in order to get grayscale of every pixels in a typical RGB pictures, we need use the formula



1. Resize the picture

Merge the four pixels into one pixel, so the total size will decrease to one fourth of the original picture. It is quite easy to implement resize function by using double “for” loops.

1. Detect the maximum and minimum brightness

In order to correct the brightness level in the picture, we need to calculate the maximum and minimum value of the brightness and adjust the brightness according to that.

1. Brightness correction

Brightness level need to be adjusted by using four threshold levels according to the maximum and minimum value of the pictures which has been detected in the former procedure.

1. Sobel edge detection

The IMG_256Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of [high spatial frequency](http://homepages.inf.ed.ac.uk/rbf/HIPR2/freqdom.htm) that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.

The operator consists of a pair of 3×3 [convolution kernels](http://homepages.inf.ed.ac.uk/rbf/HIPR2/convolve.htm) as shown in the figure.

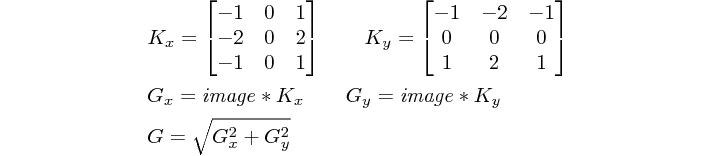


Fig 2 sobel operator

1. Output ASCIIimages

Convert the pixels into ASCII characters and the final output is displayed on the terminal

3.Platform Architecture

All processors can serve as both data master and instruction master. Components such as Performance Count Unit, Parallel Port, Interval Timer, PIO, JTAG\_UART, SDRAM Controller, SRAM Controller and On-Chip Memory, they serve Avalon Memory Mapped Slaves. On-Chip FIFO Memory and Altera Avalon Multex also serve as Avalon Memory Mapped Slaves. CPU\_0 is the data master of Performance Count Unit, Parallel Port, Interval Timer, PIO, JTAG\_UART, SDRAM Controller, SRAM Controller, On-Chip Memory, On-Chip FIFO Memory and Altera Avalon Multex. CPU\_0 is the instruction master of SDRAM Controller, SRAM Controller and On-Chip Memory.

CPU\_1, CPU\_2, CPU\_3 and CPU\_4 are the data master of On-Chip Memory, On-Chip FIFO Memory, Altera Avalon Multex and their own JTAG\_UART and Interval Timer. They are the instruction master of On-Chip Memory. The structure above is shown as Figure 3. CPU\_0 communicates with other CPUs by Altera Avalon Multex and On-Chip FIFO Memory.

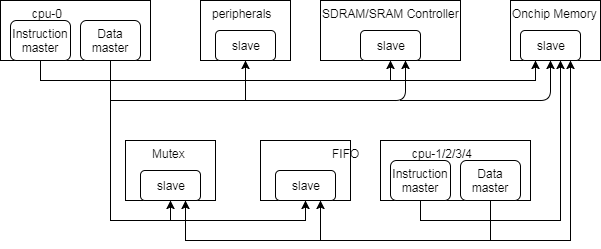


Fig 3 platform architecture

4.Component Interconnects

CPU\_0 is connected to SDRAM, SRAM, Timer\_0\_A, Timer\_0\_B, jtag\_uart\_0, switches, LEDs, HEX, buttons, performance counter, mutex\_0-4, fifo\_0-1 and shared\_onchip. CPU\_1, CPU\_2, CPU\_3 and CPU\_4 are connected to mutex\_0-4, fifo\_0-1 and their own onchip memory, timer and jtag\_uart. The component interconnections is shown as Figure4.

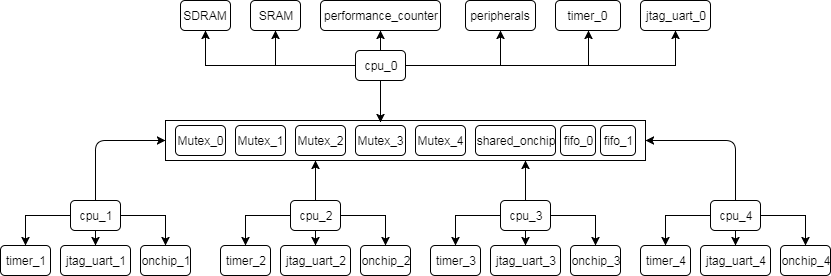


Fig 4 component interconnects

5.concurrent processes network

When implementing image processing functions, five semaphores must be used. And basic logic is that the initial value of Task1Sem is 1, so that the task1 can run at the beginning, and after task1 execution, the value of Task1Sem is reduced to 0 and the value of Task2Sem increases to 1 so that task2 can run. Same as task2, task3, task4 and task5. The concurrent process network is shown as figure 5.

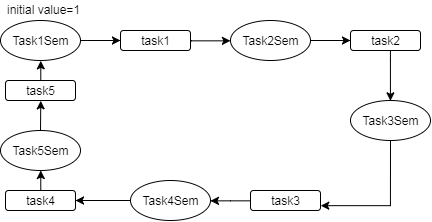


Fig 5 concurrent process network

6.measurements

|  |  |  |  |
| --- | --- | --- | --- |
|  | Single core(RTOS) | Singlecore(Bare-metal) | Multiprocessor |
| Trough put(s-1) | 0.784 | 0.796 |  |
| SRAM(byte) | 182156 | 148956 |  |